AUTOMATIC, DYNAMIC USER INTERFACE CONFIGURATION

Inventor:

Jessica Kahn

Background of the Invention

Field of the Invention

[0001] The present invention relates generally to user interfaces, and more particularly to automatically and dynamically adjusting characteristics of a user interface in response to a user's proficiency.

Description of the Related Art

[0002] User interfaces for computer systems have become increasingly sophisticated over the years. As software developers have added more and more features to operating systems and software applications, they have struggled to provide easy-to-use mechanisms for accessing and using these features. Some features are of value to experienced users but are rarely needed by novices. The increased complexity these features add to a user interface can be a significant

burden to novice users, as it can make it more difficult for such users to find the commands they are looking for when the commands are buried within complex menus containing esoteric commands.

[0003] In addition, some operations are simple when a number of default assumptions are made as to various options, but are more complex if the user needs to specify these assumptions. A novice user would rather have the software assume the default answers to the various options, while a more experienced user would prefer to have greater control over the operation by being given the opportunity to manually specify many of the options. For example, when copying files to a CD, a novice user may want the computer to simply assume standard default configuration options, while an experienced user would generally want to have more control over the copying operation.

[0004] Some operating systems and applications provide configuration options that allow a user to specify a user interface level, such as a "beginner" or "advanced." If the user selects the beginner level, he or she is presented with simplified menus and command structures, which are generally a subset of those provided at the advanced level. Other user interfaces provide techniques for customizing the menus and commands by selecting from a list of commands; selected commands are included, while other commands are omitted. One limitation of such approaches is that the user must select and/or customize the

user interface level him- or herself; many users, particularly novice users, do not know how to make such adjustments, or simply do not bother to make them.

[0005] In addition, such solutions do not allow for the fact that a user may become more proficient with time, and therefore may be ready to move to a more advanced level. Instead of providing a dynamic user mode, such solutions are essentially static, and require the user to take affirmative steps in order to change the user mode to accommodate a higher proficiency level. Again, many users simply do not bother to make such changes.

[0006] Other limitations of existing configuration schemes are:

[0007] - users often overestimate their capabilities;

[0008] - users may find it annoying to be queried as to their capabilities; and

[0009] - a perceived stigma associated with selecting a "beginner" level may cause users to inappropriately select "advanced" level.

[0010] As a result, such user interfaces are often misconfigured for the particular user. Novice users may end up with an interface that is too confusing and complex for their needs, and experienced users may become frustrated when needed commands are unavailable or difficult to access. In addition, by failing to adapt to users' changing needs and proficiency levels, existing solutions do not provide users with the ability to become more proficient by learning and discovering new features of the software. A novice user may never become aware of advanced features that are absent from the beginner version of the user

interface, and may never even become aware that an advanced version of the user interface exists.

[0011] Some user interfaces attempt to adapt to user needs by hiding menu commands that are less frequently used, and/or by changing the order of menu commands according to frequency of use. Such attempts are somewhat dynamic in nature, as they react to monitored user behavior. However, these techniques are based solely on frequency of use of commands, and fail to take into account user proficiency or other factors in deciding which menu items should be hidden or displayed. In addition, the default or initial state of such systems usually displays all menu commands, which can be overwhelming or intimidating for novice users.

[0012] What is needed is an automatic, dynamic system and method of adjusting the characteristics of a user interface based on monitored user proficiency. More proficient users could automatically be presented with a more sophisticated user interface with a broader range of commands and options, while novice users could automatically be presented with a simplified interface. What is further needed is a system and method that adapts to changing proficiency level as the user learns more about the software, and that increases the level of complexity of the user interface accordingly. What is further needed is a system and method that does not require any effort on the part of the user to

configure the user interface appropriately based on his or her level of proficiency.

Summary of the Invention

[0013] According to the present invention, markers that describe a user's actions are monitored in order to determine his or her level of proficiency. The user interface is adjusted accordingly, so that the user is automatically presented with an interface that is appropriate for his or her level of proficiency.

[0014] A set of proficiency markers relevant to user proficiency is defined. A background process provided in an application, or the operating system, or a separate utility, collects information from the markers to determine how the user is interacting with the system. The background process aggregates the collected information to determine and store a proficiency level according to the markers. Applications and/or the operating system can access the stored proficiency level so that they can modify their user interfaces accordingly. Based on the proficiency level, applications and/or the operating system can adjust the layout, organization, and command structure of their interfaces, as well as to certain elements of their content. For example, a help system could display different types of help text depending on the determined proficiency level: for novice users, basic help text might be displayed; while for experienced users, more advanced help text might be displayed.

[0015] The system and method of the present invention periodically or continually retrieves and analyzes stored markers to update stored proficiency levels, so that applications and/or the operating system can detect changes in the user's proficiency level and adjust their user interfaces accordingly. In this manner, the present invention allows the system to grow in complexity as the user's proficiency level increases, providing a technique for presenting each user with an interface that is appropriate to his or her level. As described in more detail below, the system and method of the present invention also provide commands and configuration options for partially or completely disabling the automatic user mode adjustment operation, or for restricting changes to particular areas or elements of the software.

[0016] User interface adjustment can be selectively performed based on monitored proficiency for certain aspects or elements of the software. For example, within a word processing application a user may become proficient in manipulating tables and cells, but may remain a novice with respect to mail-merge. By detecting user interaction that indicates differing levels of proficiency for different aspects of the software, the present invention could provide more complex and sophisticated commands for the tables and cells areas of the program, while retaining a simplified interface for mail-merge-related functionality.

Brief Description of the Drawings

[0017] Fig. 1 is a block diagram depicting a system for practicing the present invention according to one embodiment.

[0018] Fig. 2 is a flow chart depicting a method of updating a stored proficiency level according to one embodiment.

[0019] Fig. 3 is a flow chart depicting a method of automatically configuring a user interface according to one embodiment.

[0020] Fig. 4 is an example of a configuration dialog box for adjusting automatic configuration options according to one embodiment.

[0021] The Figures depict a preferred embodiment of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Detailed Description of the Invention

[0022] Referring now to Fig. 1, there is shown a block diagram depicting a system for practicing the present invention according to one embodiment. The various functional components shown therein can be implemented, for example, in a conventional personal computer such as a Power Macintosh G4 running an operating system (OS) 101 such as MacOS X (both available from Apple

Computer, Inc., of Cupertino, California), along with one or more applications 102. For illustrative purposes, three applications 102 are shown in Fig. 1, although one skilled in the art will recognize that any number of applications 102 may be running on the computer system at any given time. In addition, one skilled in the art will recognize that the present invention can be implemented in other contexts, such as for example in an environment for running web-resident applications from a client machine (provided, for example, by an application service provider).

[0023] Background process 103, such as a daemon, periodically retrieves and reads markers 105 to determine an estimated user proficiency level 107. Background process 103 then stores a representation of the user proficiency level 107 in library 104 that is accessible to applications 102 (and to operating system 101) via an application programming interface (API) 106. Each application 102 and/or operating system 101 can then configure its user interface according to the determined proficiency.

[0024] Background process 103 can be implemented as part of operating system 101, or it can be a plug-in or other type of application. In an alternative embodiment, the process for monitoring user proficiency can be incorporated into the application whose interface will be modified without support from the operating system or other system components. In general, it is preferable if background process 103 runs without interfering with the user's experience in

operating the computer. In one embodiment, background process 103 runs continuously to generate and update user proficiency level 107 in library 104. In another embodiment, background process 103 runs periodically (such as daily or weekly), or upon occurrence of a specified event (such as system startup, application launch, or the like).

[0025] Markers 105 can represent any facet of user interaction with an application 102 or operating system 101, and can include any available information describing or summarizing behavior or usage patterns. Any number of markers 105 can contribute to an estimate of user proficiency; the particular contributing markers 105 are selected based on the degree to which they accurately indicate proficiency. When more than one marker 105 contributes to the proficiency determination, the relative contribution of the markers 105 can be determined based on predefined relative weights. In one embodiment, markers 105 correspond to (and/or are derivable from) data that is already regularly stored by an application 102 or operating system 101, so that there is no additional burden involved in collecting relevant data for proficiency estimation. In another embodiment, some or all of the markers 105 are specially derived for the purpose of determining user proficiency; thus, they represent a summary or description of monitored user behavior.

[0026] For example, if a user interface is to be configured based on user proficiency in a specific aspect of a word processing application interface, in one

embodiment an API would set up the markers to be monitored, and would read the markers to determine a proficiency level in the specific aspect of interest. The operating system could monitor a set of markers that are available to it, and could further provide access to proficiency levels for an application or aspect of an application based on the monitored markers. Applications, plug-ins, and other software components could set new markers using the same API, and further could categorize the markers and extract proficiency levels by reading particular markers of interest. In this manner, applications can determine overall proficiency, application proficiency, and/or proficiency for a particular aspect of an application.

[0027] Library 104 contains some representation of user proficiency level 107 that can be read by applications 102 via API 106. The representation of user proficiency can be an overall measure, or it can be a representation of a marker 105 or combination of markers 105. In one embodiment, library 104 contains representations of several proficiency levels 107. For example, different proficiency levels may apply to different applications 102, or to different aspects of a particular application 102. This reflects the fact that a user may become adept at some applications 102 while remaining a novice with respect to other applications 102; or the user may become proficient at some features of an application 102 but not others. Each proficiency level can therefore be determined using a different set of markers 105, depending upon a

determination as to which types of behavior and/or statistics tend to indicate proficiency level for the particular application or feature set within an application. API 106 provides the appropriate interface to allow application 102 to retrieve the relevant proficiency level representation 107 for that application 102 or for the portion of application 102 being configured.

[0028] For example, in an e-mail application, a user who has configured a number of e-mail accounts might be deemed more proficient than one who has only used a single default account. Thus, the number of e-mail accounts would be used as a marker 105 that, combined with other markers 105, provides an estimate of user proficiency for an e-mail application. A marker indicating the number of e-mail accounts is readily available from existing information that is used by the application 102 in its normal operations. An e-mail application 102 could therefore obtain, via library 104, a proficiency level 107 based on a marker 105 that indicates the number of e-mail accounts configured by a user, combined with other markers 105.

[0029] Other examples of markers 105 that could be used to determine user proficiency include without limitation:

[0030] - number and organization of bookmarks in a Web browser application;

[0031] - degree to which certain system features have been customized by the user;

[0032] - number and types of peripherals connected to the system;

[0033] - number of applications being run concurrently (either current value or historical average);

[0034] - total system uptime; and

[0035] - whether or not the user has made use of certain advanced features.

[0036] Combinations of markers 105 can also be used. A proficiency level 107 can be derived by adding, averaging, or otherwise combining the values of two or more markers 105, weighted appropriately. Alternatively, a score based on two or more markers 105 can be determined, and proficiency level 107 is determined by comparing the score with particular threshold levels. For example, a score can be determined by calculating:

[0037] $S = U + 10F + 8A$,

[0038] where U = total system uptime in days;

[0039] F = number of features customized by the user; and

[0040] A = average number of applications run concurrently.

[0041] A user having a score S exceeding 100 might be given the highest proficiency level 107; a score between 50 and 99 might correspond to medium proficiency level 107; and a score below 50 might correspond to beginner proficiency level 107.

[0042] In addition, for computer systems used by more than one user, each user may have a profile. Library 104 contains proficiency level 107 for each user,

and provides access via API 106 to proficiency level 107 for the user that is currently logged in to the system.

[0043] In one embodiment, API 106 is available to applications 102, operating system 101, and/or other user interface level code within the computer system, such as OS libraries for providing user interfaces. Examples of such libraries include AppKit and High Level Toolbox, both of which are provided in MacOS X available from Apple Computer, Inc., of Cupertino, California.

[0044] Applications 102 and OS 101 use the proficiency level 107 from library 104 to configure their user interfaces to match the determined proficiency of the user. Examples of such configuration include without limitation:

[0045] - enabling or disabling certain commands, menu options, keyboard shortcuts, and the like;

[0046] - tailoring content, such as Help text, to match the proficiency level;

[0047] - changing the appearance of certain user interface elements (for example, changing the sizes of icons, or adding or removing text labels);

[0048] - enabling or disabling user help features such as on-screen tips and assistance; and

[0049] - enabling or disabling certain configuration options.

[0050] In one embodiment, users can enable or disable the automatic configuration scheme via a preferences or options screen, and can override the automatic configuration if desired. In one embodiment, users can enable certain

types of automatic configuration while disabling other types. For example, a user can specify that menus, ToolTips, and Help text be automatically configured, but that Toolbars not be automatically configured. Such a selection is depicted in Fig. 4.

[0051] In one embodiment, applications potentially reconfigure their interfaces on a periodic basis, or upon startup, or upon occurrence of some other event. For example, an application 102 might check library 104 each time the application 102 launches, to determine whether the user's proficiency level 107 has changed, thus warranting a reconfiguration of the user interface. Alternatively, such checking may take place on a weekly basis, or upon user request, or at any other time. In one embodiment, when a change to a user interface is made, the user is alerted to the change, particularly when new features are added. The user can be given an opportunity to view a "What's New" screen or other description of the newly-added features, so that he or she can be made aware of the new functionally and thereby make better use of the system. In one embodiment, the user is given an opportunity to decline any new feature that is added, thereby overriding the system's determination that the new feature should be added.

[0052] The present invention thus provides a mechanism by which user proficiency can be discovered automatically and unobtrusively, and by which user interface features can be automatically configured to match the determined

proficiency. The interface thus grows in complexity as the user's proficiency level increases. This allows all users, from novices to experts, to make more efficient use of the software applications 102, operating system 101, and the computer system as a whole.

[0053] Referring now to Fig. 2, there is shown a method of updating a stored proficiency level according to one embodiment. Operating system 101 initiates 202 background process 103 for updating proficiency levels; this may occur, for example, upon system startup, or upon user activation, or upon some trigger event. Then, the user runs 203 applications 102 normally. As applications 102 are run, they store and update various data items that are identified as markers 105. As described above, such markers 105 may be information that is normally stored by applications 102, information that is specifically stored for purposes of assessing user proficiency, or a combination of both.

[0054] Background process 103 retrieves 204 stored markers 105 and combines 205 them to determine a proficiency level 107. As described above, proficiency level 107 can be specific to a particular application 102 or to OS 101, or it can be specific to particular features or components of an application 102, or it can apply to all applications 102. Background process 103 updates 206 proficiency level 107 in library 104 so that applications 102 can retrieve relevant proficiency level information 107 via API 106.

[0055] In one embodiment, background process 103 performs steps 204 through 206 periodically. In another embodiment, background process 103 performs steps 204 through 206 in response to user activation, or in response to a triggering event such as startup, launch of an application 102, or the like. If background process 103 determines 207 that proficiency level 107 should be re-evaluated (for example, upon occurrence of a triggering event), steps 204 through 206 are repeated.

[0056] The above description sets forth an example of a method of updating proficiency levels 107. The method is described in terms of the operation of a background process 103. However, one skilled in the art will recognize that the steps of Fig. 2 may be performed by any computer-implemented process, software, or system, and need not be performed by a background process 103. For example, the steps may be performed by a foreground application that runs upon startup.

[0057] Referring now to Fig. 3, there is shown a method of automatically configuring a user interface according to one embodiment. The method of Fig. 3 may be performed, for example, by applications 102 or by OS 101 via API 106. As described above, API 106 contains the function calls and other appropriate modules for accessing proficiency levels 107 stored in library 104. Applications 102 or OS 101 may perform the steps of Fig. 3 periodically, or in response to user activation, in response to a triggering event such as startup, launch of an

application, or the like. In one embodiment, application 102 can register a callback with library 104, so that when certain markers 105 change, library 104 notifies application 102 (via the callback) that a proficiency level update may be warranted. One possible triggering event is the detection of a change to proficiency level 107 in library 104; thus an application 102 might perform the method of Fig. 3 when step 206 of Fig. 2 has been performed. Alternatively, the method of Fig. 3 can be performed independently of the steps of Fig. 2.

[0058] Proficiency level 107 is retrieved 302 from library 104. In one embodiment, where multiple proficiency levels 107 exist for different applications 102 and/or OS 101, only the relevant level 107 is retrieved. Based on the retrieved proficiency level 107, the user interface configuration is changed 304. The particular changes made to the user interface configuration can vary from application 102 to application 102. As described above, such changes may include enabling or disabling commands, changing help text, and the like.

[0059] If further UI configuration is needed 305, for example upon detection of a triggering event, steps 302 and 303 are repeated.

[0060] In one embodiment, the user is given an opportunity to control or override the automatic user interface configuration. For example, an application 102 or OS 101 may provide a command for accessing an "options" or "preferences" dialog box that allows the user to indicate whether or not automatic configuration should be enabled, and which UI features should be

automatically configured. In one embodiment, the configuration options apply to all applications 102 and OS 101; in another embodiment, the configuration options apply to a specific application 102 or OS 101.

[0061] Referring now to Fig. 4, there is shown an example of a configuration dialog box 400 for adjusting automatic configuration options according to one embodiment. Dialog box 400 may be displayed, for example, in response to a user selecting an "options" or "preferences" command from within application 102 or OS 101.

[0062] Checkbox 405 enables and disables the automatic UI configuration functionality described above. When checkbox 405 is unchecked, applications 102 do not perform automatic UI configuration. This may be enforced, for example, by having the API return a NULL or "no value" response when applications 102 query it for proficiency level, thus causing automatic UI configuration to be disabled. In one embodiment, when a user unchecks checkbox 405, the UI configuration currently in place is maintained; in another embodiment, the UI configuration reverts to a default configuration or to a user setting.

[0063] Checkboxes 401 provide controls for enabling and disabling the automatic UI configuration functionality with respect to particular areas of applications 102. For example, a user could indicate that he or she wishes to

enable automatic configuration for menus, ToolTips, and help text, but not for Toolbars (as shown in Fig. 4).

[0064] Default button 402 reverts to a default setting. OK button 402 accepts the user's changes and dismisses dialog box 400. Cancel button 402 cancels the user's changes and dismisses dialog box 400.

[0065] One skilled in the art will recognize that the particular arrangement and items shown in Fig. 4 are merely exemplary, and that many other arrangements of dialog box 400 may be contemplated without departing from the essential characteristics of the present invention. For example, the configuration options for adjusting automatic configuration options may be provided within the context of a dialog box that also contains other configuration options related to other aspects of the application 102.

[0066] As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The particular architectures depicted above are merely exemplary of one implementation of the present invention. The functional elements and method steps described above are provided as illustrative examples of one technique for implementing the invention; one skilled in the art will recognize that many other implementations are possible without departing from the present invention as recited in the claims. Likewise, the particular capitalization or naming of the modules, protocols, features,

attributes, or any other aspect is not mandatory or significant, and the mechanisms that implement the invention or its features may have different names or formats. In addition, the present invention may be implemented as a method, process, user interface, computer program product, system, apparatus, or any combination thereof. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.